Irrigation System

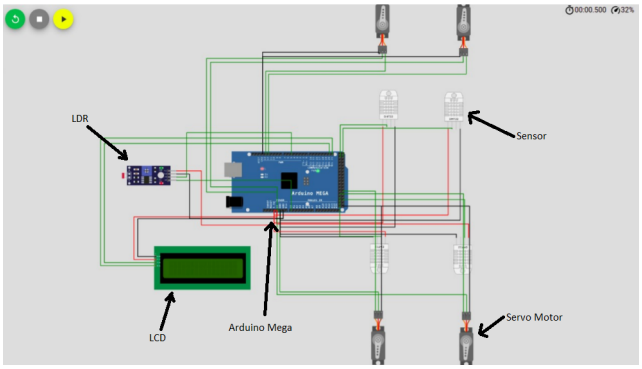
Harishchandra Patidar, Ravi Shankar Das*
Roll No: 21111029,21111052
CS698T- 2021-22 (first semester)
Indian Institute of Technology,
Kanpur, INDIA-208 016

November 3, 2021

1 System Design

For the desigining of the circuit we have used the following components:

1. Arduino Mega which is powered by the ATmega2560 chip, has 256K bytes of Flash memory, 8k bytes of SRAM and 4K bytes of EEPROM. The board features 54 digital pins, 16 analog input pins, and 4 serial ports. It runs at 16MHz.
2. DHT22 Digital Temperature and Humidity Sensor This sensor reads both value temperature and humidity values.
3. Photoresistor (LDR) sensor module This sensor is used for day and night time detection . The photoresistor sensor module includes a LDR (light-dependant resistor) in series with a 10K resistor. The AO pin is connected between the LDR and the 10K resistor.The voltage on the AO pin depends on the illumination - that is the amount of light that falls on the sensor.
4. Standard Micro Servo Motor is used for controlling the water flow.
5. Lcd1602 which has 2 lines, 16 characters per line . This will be used to display the percentage of water flow by each motor.



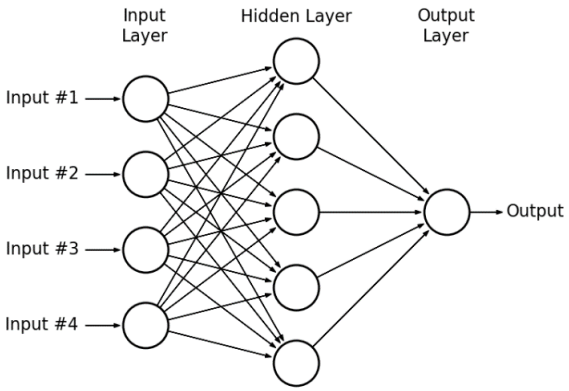
*E-mail: : hpatidar21@iitk.ac.in, ravisd21@iitk.ac.in

1.1 Functionality of the circuit

Data of temperature and humidity for field is collected using DHT22 sensor , we have design a function for water percentage calculation , we gave this temperature and humidity as input to this function and we get water percentage as output and this percentage will be printed on LCD screen after this we convert percentage into degree (0-180) and gives input to servo motor also there is one more sensor called LDR which count the light intensity and based on the value of this senor we find out , is there day time or night time . If its night time we set water percentage as ZERO else it will work as normal. Here is the link for the simulator.

2 ML Model

In Neural Networks, there are three standard models that are used to make predictions which are Multilayer Perceptrons (MLP), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). In this assignment we have used the Multilayer Perceptrons(MLP), which is a standard fully connected neural network model. It is comprised of layers of nodes where each node is connected to all outputs from the previous layer and the output of each node is connected to all inputs for nodes in the next layer[3].In each layer there are processing units called as perceptrons which closely resembles the human nervous system's neuron.

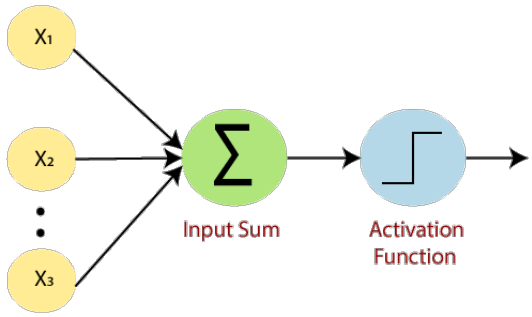


Source: http://stackexchange.com/

The Above picture is a figure of MLP model consisting of one input layer , one hidden layer and an output layer and as we can see every layer is fully connected to the next layers node. In this assignment we will be using Tensorflow and keras open-source library for performing the ML task and Google Colab for running the model where our task is to perform regression which is defined as the predicting an output y given one or several inputs to the model, we basically set up the relationship between the inputs and the outputs so that when new data comes the model can predict their outputs. These are the steps which were followed to make the model:

1. **Defining the model**, in this step we need to choose the type of the model, number of perceptrons(neurons) per layer and the activation function. For the

type of the model we have two choices namely **sequential or Functional** model, in this assignment we will be using the sequential model where there is a linear path among the various layers. Now for the number of nodes we have tried running the model for various number of perceptrons and found which number was giving a good model using the evaluation metric of **Mean absolute error**. For the activation function we have used the **ReLu function**,which is given as f(x) = max(0,x). When the inputs are taken the weighted sum is calculated and bias is added after which they are passed through an activation function which helps in deciding if the perceptron should get activated or not. Below figure is the figure of perceptron where x's are the inputs they get multiplied by the weights and the weighted sum is passed through the activation function.



Source: https://www.javatpoint.com/

2. **Compiling the model**, in this step there are three things first is the optimizer which as the name suggest optimizes the model to increase the accuracy , we have used the most common **ADAM** optimizer. Second is the loss which is also used by the model for optimization which reduces the error here we have used **Root mean square error (RMSE)** as the loss function which is root of square of the difference between the actual and the predicted value and the last one is the evaluation metric upon which the model will be judged, Now the metric type depends upon the nature of the model , which is for regression MAE or MSE or RMSE is used while for classification accuracy metric is used, since our model is for regression we have used the **Mean absolute error(MAE)** MAE can also be understoof as if your MAE is x it means, that if you randomly choose a data point from your dataset, then, you would expect your prediction to be x away from the true value.

3. **Training the model**, in this step we train/fit the model, first we split the dataset into train and test set and train on the train set. Number of times the model iterates through the train set in ML terms is called as Epoch, One more term used is the batch size which is the size of sample taken in every iteration to train the model and learn the optimal parameters and update the model.

4. **Evaluating the model**, after we have trained the model we need to check how our model is performing on new unseen data that we have set aside(test set). There are two ways of evaluating the model, first one in evaluate method provided by tensorflow which tells us the overall error and MAE new unseen test

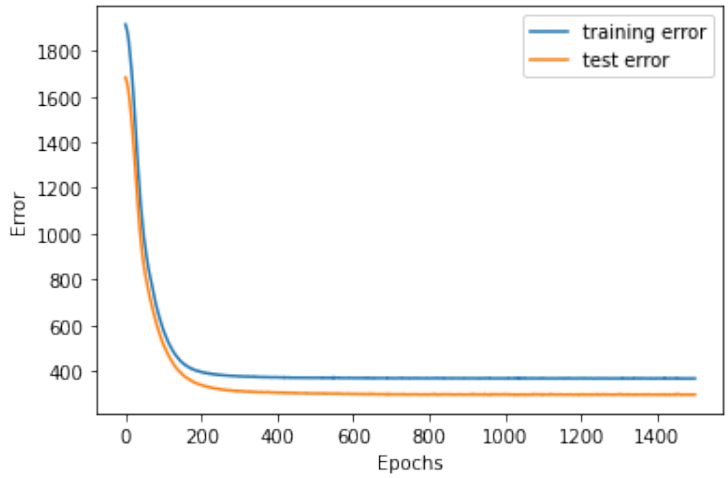
set and another one is looking at the plot of training and testing error , check if the model is not improving.

2.1 Optimization

For optimization there are several methods namely Early stopping , Dropout, Regularization and Simpler Architecture. I tried all of them but Simpler architecture was giving the best results , in this method we try to decrease the model complexity by decreasing the number of layers and number of perceptrons per layer.

2.2 Results

Since we cannot decrease the number of layers for the number of perceptrons per layer i used the brute force to find the parameters which gave the least Mean Absolute Error. The brute force was runing for more than 7 hours so after runnning for several times it gave the values of **1500** for the numbers of epochs , **8** for the batch size , **6** and **2** for the number of first and layer perceptron numbers. All of these parameters gave the result of MAE of ~ **11.78** and the corresponding training vs testing plot was as belows where we can see that for 200 epochs both the errors nearly co-incided and kept decreasing after which the they decreases slowly. For our best model we have taken the trained weights and the bias and put them in the source file of arduino for the prediction of water flow.



References

[1] https://docs.wokwi.com/

[2] https://stackoverflow.com/

[3] https://machinelearningmastery.com/

[4] https://www.tensorflow.org/tutorials/keras/regression